

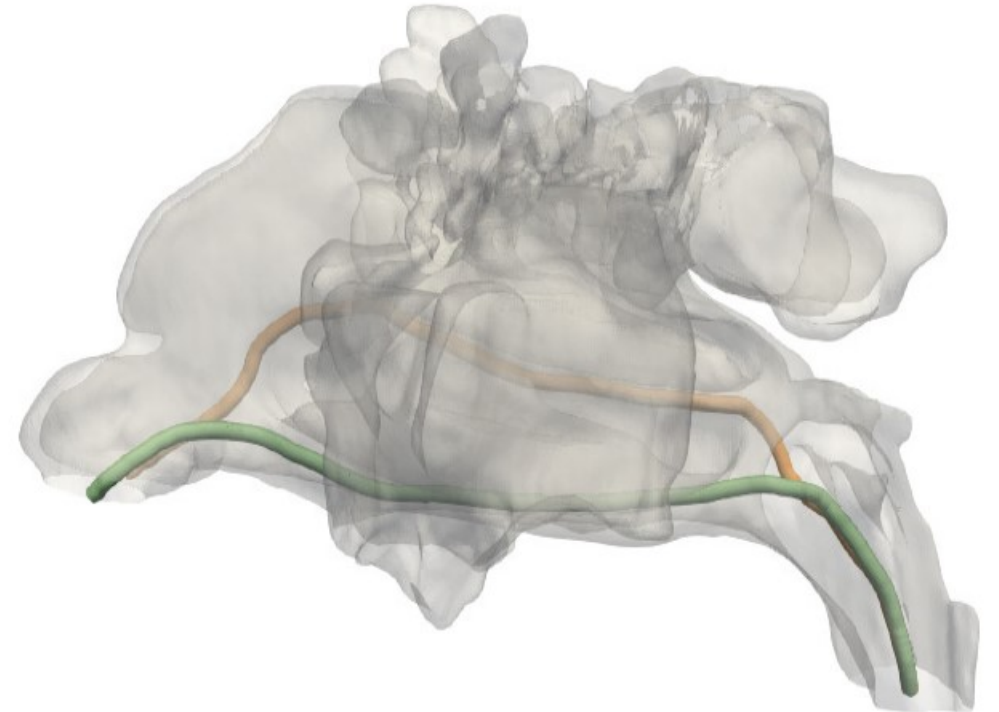


ENABELING WEB-BASED INTERACTIVE HPC FOR RHINOLOGY

2020-12-09 | CHRISTIAN WITZLER (C.WITZLER@FZ-JUELICH.DE), ALICE GROSCH (A.GROSCH@FZ-JUELICH.DE),
JENS H. GÖBBERT (J.GOEBBERT@FZ-JUELICH.DE), TIM KREUZER (T.KREUZER@FZ-JUELICH.DE)

OUTLINE

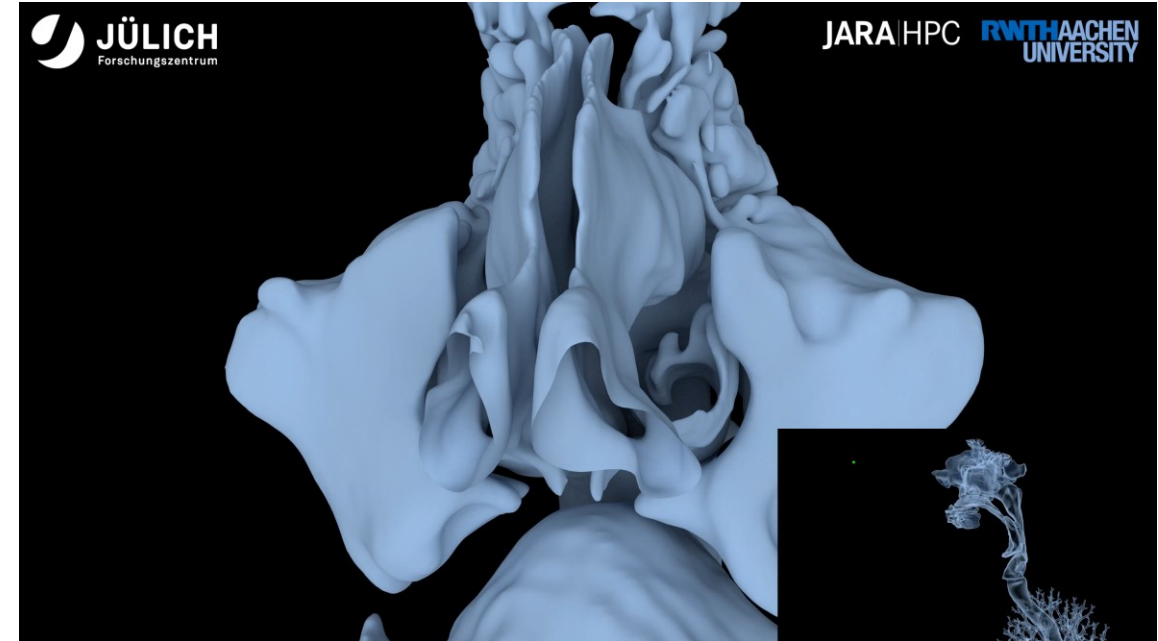
- Motivation
- Enabling interactive Supercomputing
 - Jupyter, Jupyter-JSC
 - Visualization in the browser
 - Security
- Dashboard for Rhinology
 - Development
 - Workflow management
 - Demonstration



MOTIVATION

Why do we need HPC for interactive visualization?

- Highly resolved direct numerical simulations (DNS) are essential for more insights into detailed flow properties
 - This results in extremely large data sets
 - Download can easily become impossible



DNS results must be visualized on site!

MOTIVATION

HPC at JSC

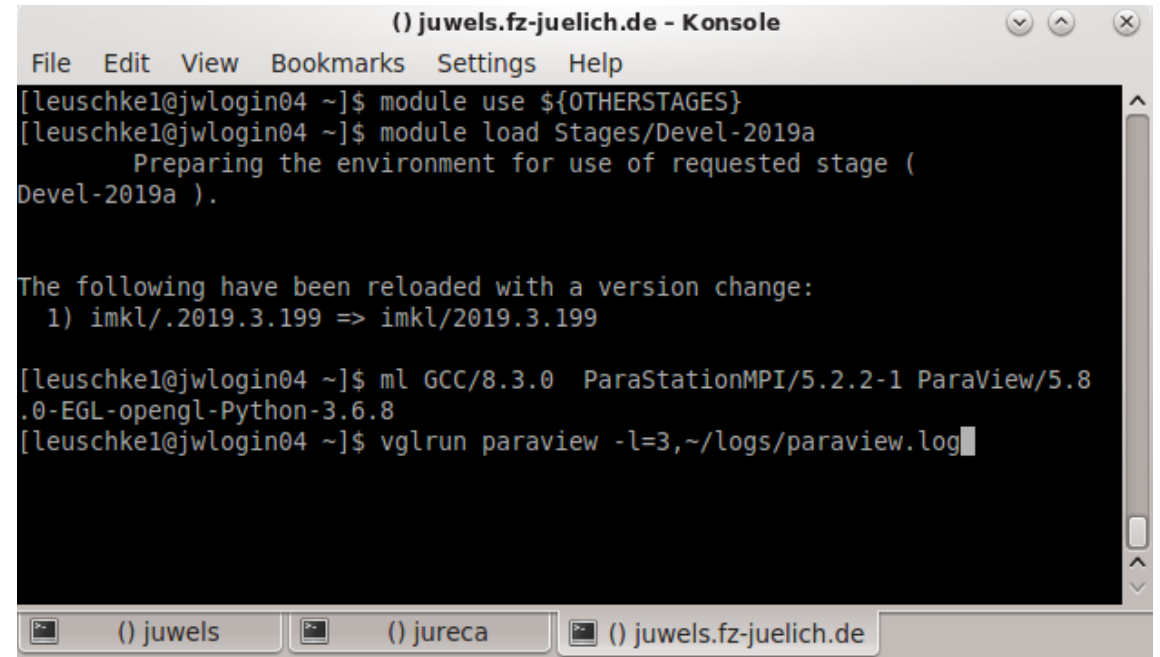
- JUWELS booster
 - November 2020
 - #7 in top 500
 - 479TB main memory
 - 71 Petaflops/s
 - 44,928 CPU cores
 - 12,939,264 CUDA cores



MOTIVATION

How we access HPC normaly?

- Normal access is via the command line
- For interactive visualization this means
 - Sign in using your ssh key
 - Start a vncserver
 - Forward the port
 - Connect a vncviewer
 - Start a visualization tool



```
( ) juwels.fz-juelich.de - Konsole
File Edit View Bookmarks Settings Help
[leuschkel@jwlogin04 ~]$ module use ${OTHERSTAGES}
[leuschkel@jwlogin04 ~]$ module load Stages/Devel-2019a
    Preparing the environment for use of requested stage (
Devel-2019a ).

The following have been reloaded with a version change:
  1) imkl/.2019.3.199 => imkl/2019.3.199

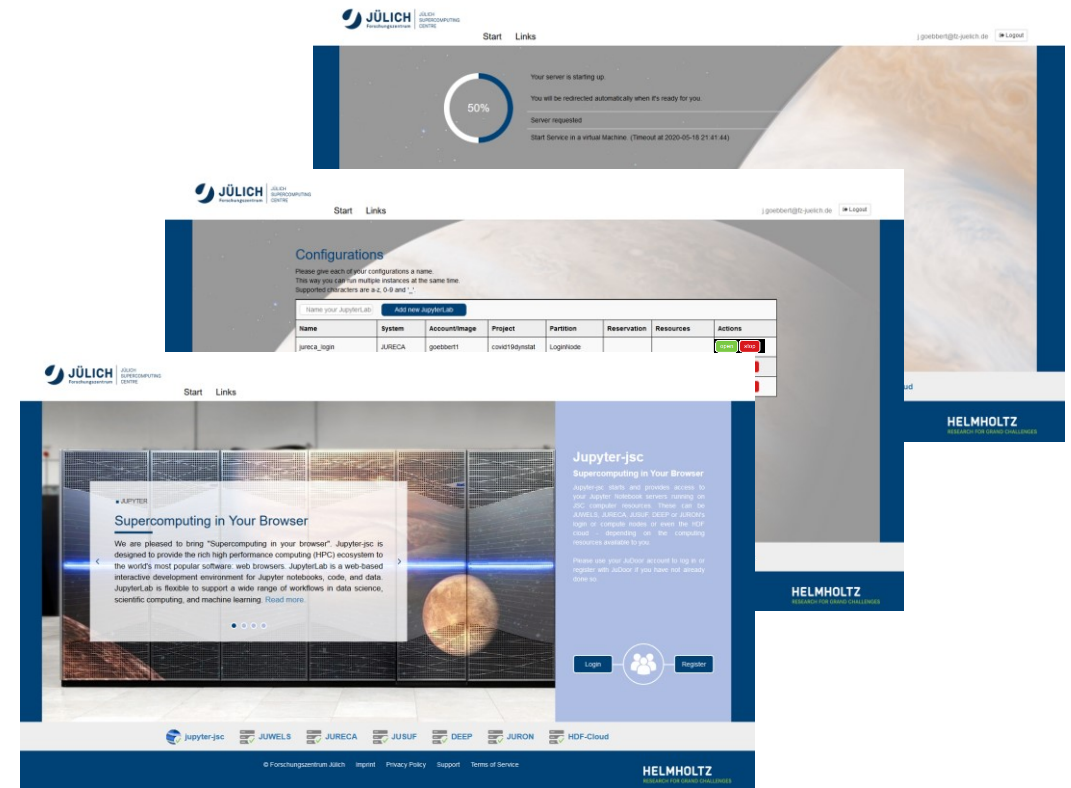
[leuschkel@jwlogin04 ~]$ ml GCC/8.3.0 ParaStationMPI/5.2.2-1 ParaView/5.8
.0-EGL-opengl-Python-3.6.8
[leuschkel@jwlogin04 ~]$ vglrun paraview -l=3,~/logs/paraview.log
```


INTERACTIVE SUPERCOMPUTING

How Rhinodiagnost enabled HPC in the browser

“It is all about using and building a machinery **interface** **between** computational researchers and data, supercomputers, laptops, cloud **and** your thinking, your reasoning, your insides, your ideas about a problem.”

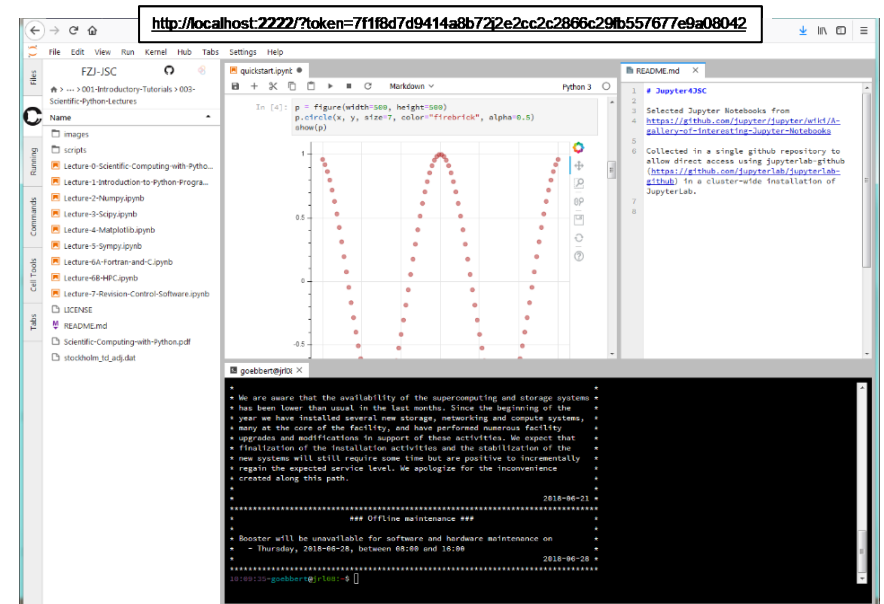
Fernando Perez, Berkely Institute for Data Science
Founder of Project Jupyter



INTERACTIVE SUPERCOMPUTING

Jupyter Notebooks

- Visualization in JupyterLab
 - Bokeh, Plotly, Matplotlib
- Remote Desktop
- But:
 - Server-side rendering
 - 3D interactive visualization
 - Inside a Jupyter Notebook



We made this possible with PVlink

INTERACTIVE SUPERCOMPUTING

PVlink

Jupyter for interactive In-Situ Visualization With ParaView/Catalyst

Alice Grosch¹, Christian Witzler¹
¹ Jülich Supercomputing Centre, Jülich Forschungszentrum GmbH, Germany



Introduction

- ParaView**^[1]: An open-source multiple-platform application for interactive, scientific visualization. Designed for data parallelism on shared-memory or distributed-memory multicomputers and clusters.
- Catalyst**^[2]: An in-situ use case library. Part of the ParaView framework dedicated to in-situ analysis.
- pvlink**^[3] using **ParaViewWeb**^[4]: A Jupyter Notebook extension leveraging the ParaViewWeb remote rendering capabilities. Starts a ParaViewWeb server process on the backend which captures a RenderView and pushes the images to a Notebook output cell. The output is fully interactive.
- Jupyter**: interactive in-situ visualization in Jupyter

Technical Overview

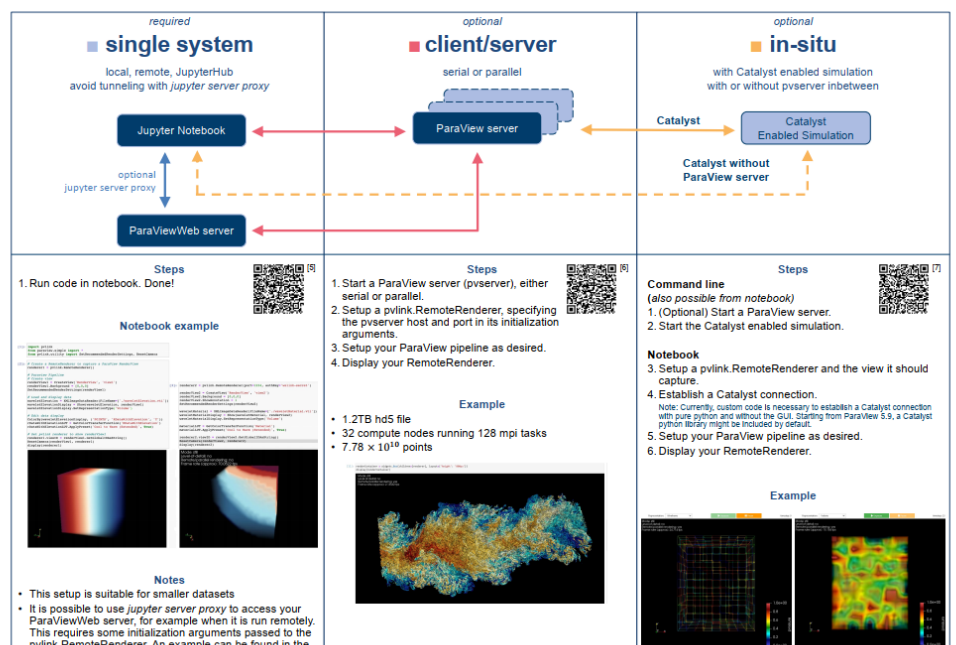
How does it work?
 The Jupyter Notebook extension pvlink is based on ParaViewWeb. On the Javascript frontend side, a ParaViewWeb RemoteRenderer renders the images it receives from the backend into the output cell of the notebook. On the Python backend, a ParaViewWeb server runs, which captures a RenderView and pushes the images to the frontend. Front and backend communicate over a websocket connection.
 Because the backend runs on Python, we can steer the entire ParaView pipeline from within the notebook. This not only includes creating views and displaying sources in them, but also the possibility of connecting to a pvserver and/or a Catalyst enabled simulation. In the last case, we will have a fully interactive in-situ visualization right in the notebook!

How secure is it?
 You can fully control the server setup. pvlink by default offers a bare minimum of security by automatically generating a random password for you, although you can choose to set your own password. A successful websocket connection can then only be established with the correct password. However, you can also choose to run your websocket over the encrypted WSS (WebSockets over SSL/TLS) protocol.

How does it scale?
 pvlink itself only takes care of rendering the images into your notebook output cell. Minus the small amount of time it takes to transfer image and interaction data, the scaling is identical to ParaView's innate scaling capabilities. In concrete terms, ParaView still handles all calculations and rendering operations, pvlink only displays the results in the Notebook output cell for you. That means that even extremely large scientific datasets can be shown and interacted with from the Notebook.

How do I use it?
 You do not need to modify your existing ParaView setup. Simply install the

Possible Setups



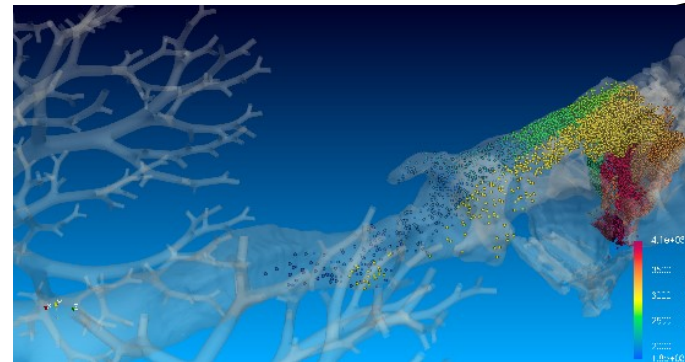
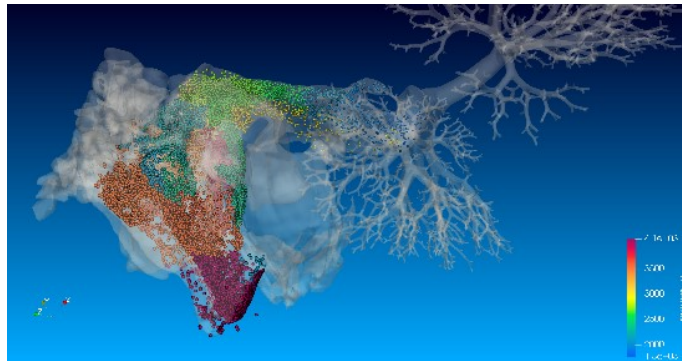
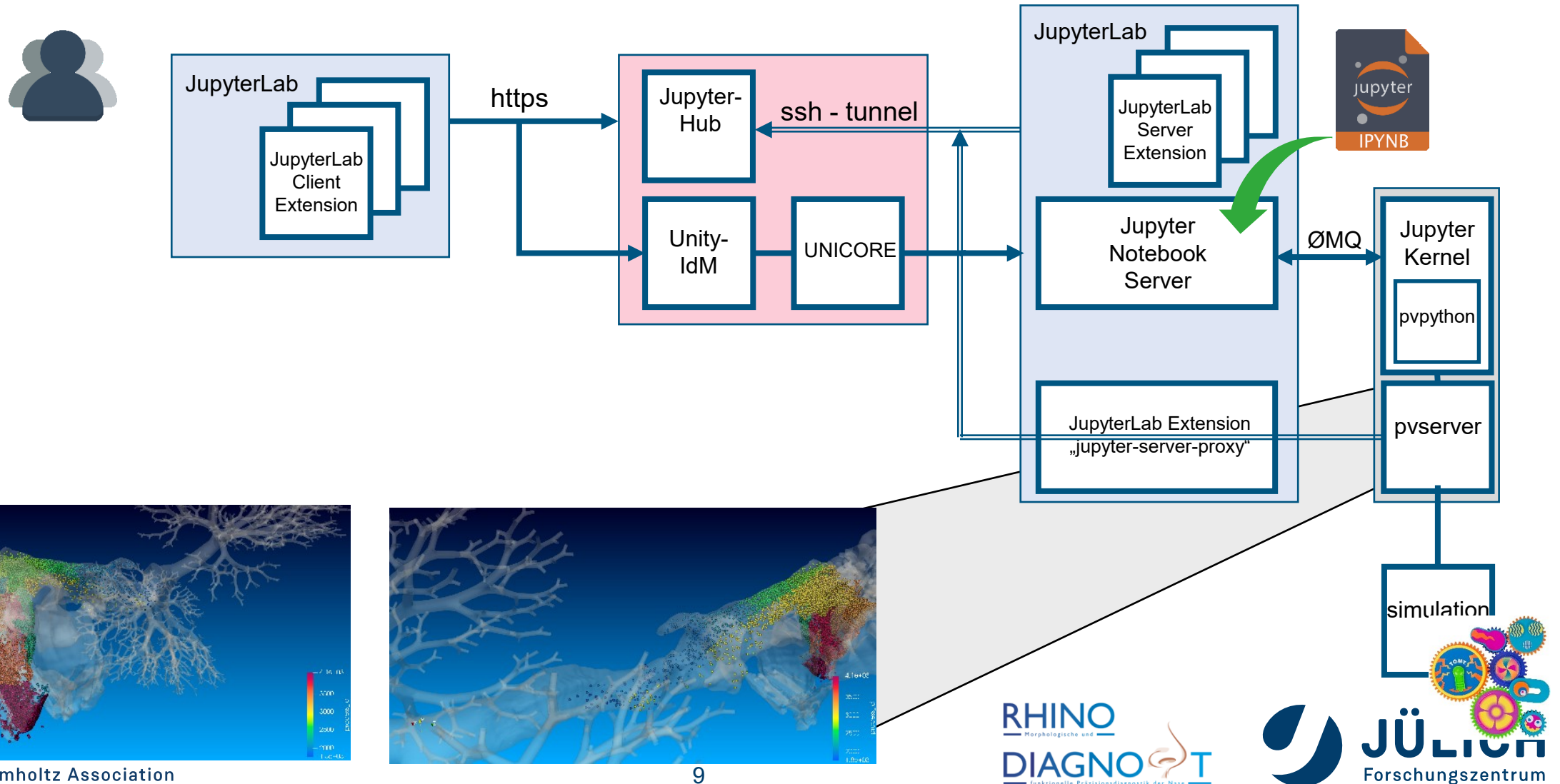
<https://cfp.jupytercon.com/2020/schedule/presentation/123/jupyter-for-interactive-in-situ-visualization-with-paraviewcatalyst/>

PVlink enables server-side 3D interactive visualization inside a Jupyter Notebook



INTERACTIVE SUPERCOMPUTING

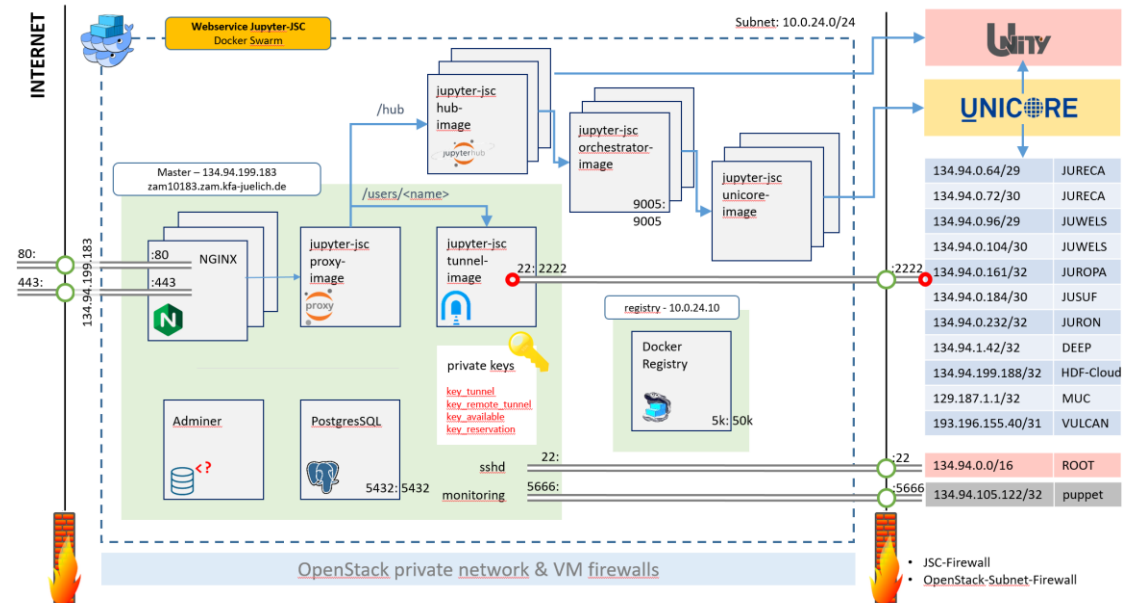
How Jupyter-JSC enabled HPC in the browser



INFORMATION SECURITY

Authentication, Authorization, Encryption, Anonymization

- Authentication – identity check
 - Registration & Login
 - Development of JupyterHub-AuthModule
- Authorization – permission check
 - Development of JupyterHub-Spawner
- Encrypted & Secured Communication
 - Outside JSC: HTTPS + session-tokens
 - Inside JSC: ssh-tunneling + password
- Anonymization
 - DICOM uploader



https://gitlab.version.fz-juelich.de/jupyter4jsc/j4j_extras/dicom-upload

DASHBOARD

Rhinodiagnost UI for HPC workflows



Projects

Geometry

Simulations

Analysis

Projects

OVERVIEW NEW

Search

	Not possible (yet)	Possible	In progress	Possible and done	Error			
	Date	Name	Description	Dicom	4PR	Geometry	Simulation (last active)	Actions
✓	08.09.2020	Example	Example project	✓	○	✓	✓	🗑️
✗	08.09.2020	d3058912f8a416b663dbc846e93fc215	Project2	✓	○	○	✗	🗑️

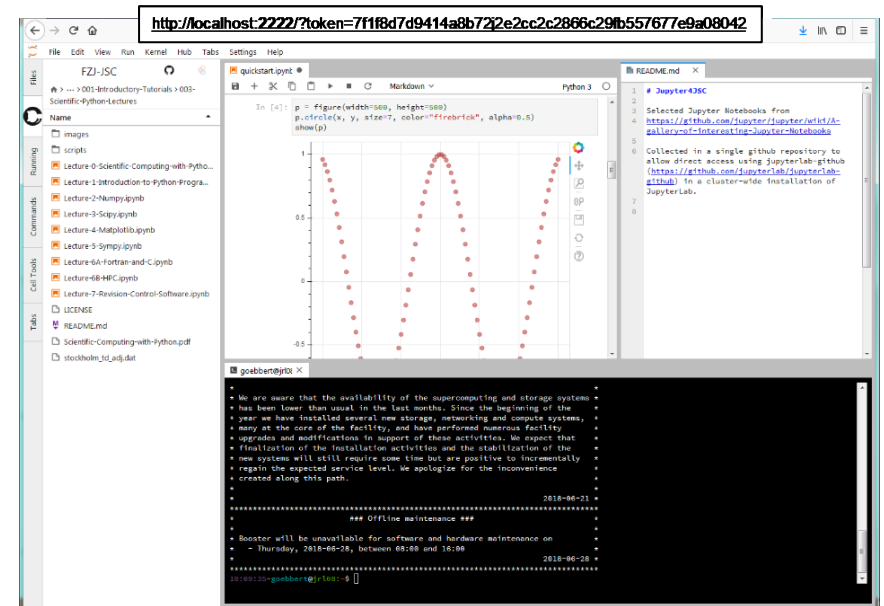
Rows per page: 5 1-2 of 2



DASHBOARD

Development - Design decisions

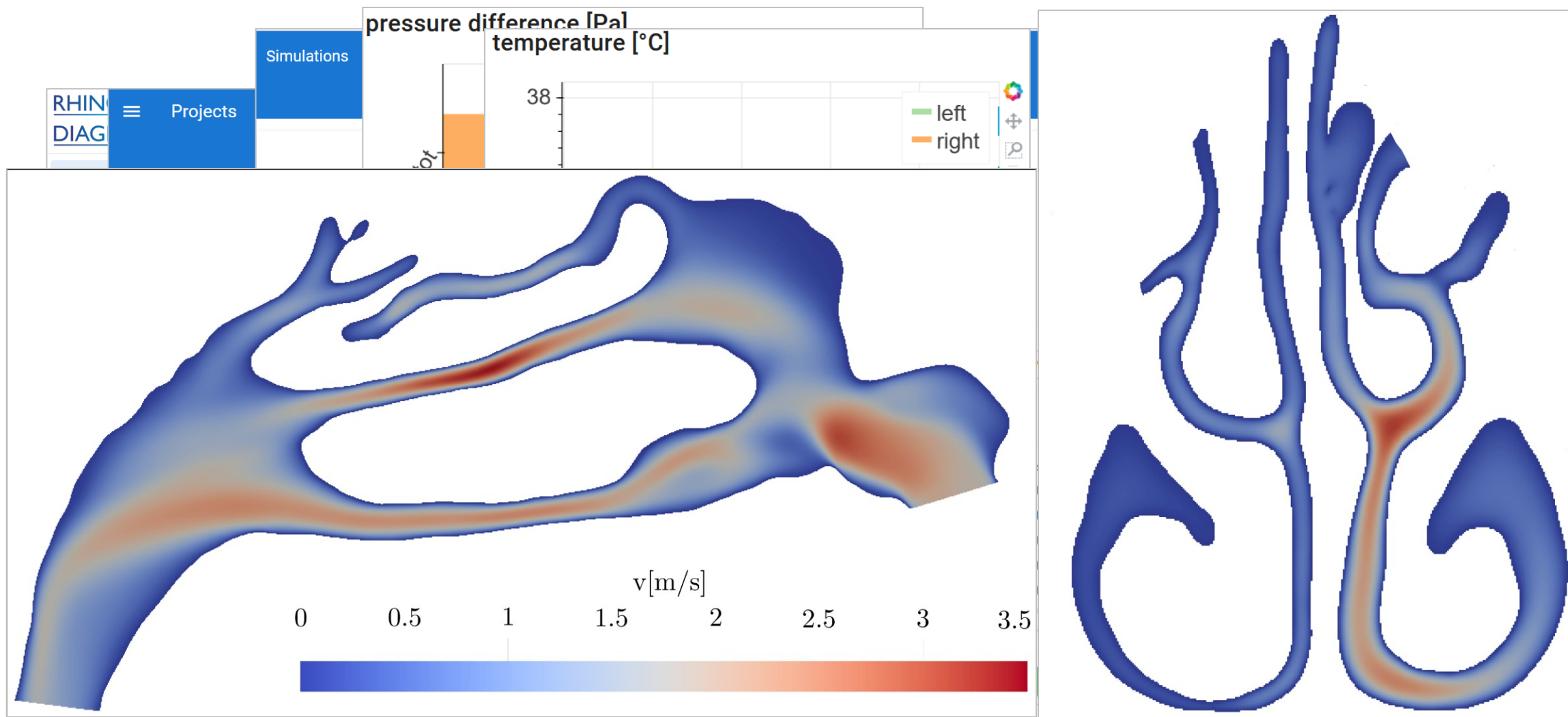
- Jupyter-JSC is backbone
- Programming language - Python
- Dashboard text & support
- With Jupyter for data analysis
- Over the Jupyter functionality
- Open source



Dashboard development with „Voilà“

<https://github.com/voila-dashboards/voila>

DASHBOARD



SUMMARY

- Motivation: highly resolved DNS must be visualized on site
- For non-domain experts, browser-based access is essential
 - Rhinodiagnost development
 - **JupyterLab through Jupyter-JSC**
- Server-side 3D interactive visualization is required
 - Rhinodiagnost development
 - **PVlink inlines access to ParaView server**
- Expressive, easy to understand, intuitive user interface for non-domain experts
 - Rhinodiagnost development
 - **Dashboard with volià based on Jupyter-JSC**

QUESTIONS?

Acknowledgments

Thanks to all project partners:

- AIT Angewandte Informationstechnik
Forschungsgesellschaft mbH
- Sutter Medizintechnik GmbH
- Rheinisch-Westfälischen Technischen Hochschule
(RWTH) Aachen
- Med Contact GmbH

Special thanks go to:

- Alice Grosch
- Bernd Schuller
- Jens Henrik Göbbert
- Mario Rüttgers
- Moritz Waldmann

This work is supported by the Rhinodiagnost project funded by the Zenrale Innovationsprogramm Mittelstand (ZIM) of the Federal Ministry of Economical Affairs and Energy (BMWi)

